LOW-RANK APPROXIMATE SINGULAR VALUE DECOMPOSITION AND IMAGE COMPRESSION

LUCAS DE LA BROSSE, ANURAG MIGLANI, THOMAS ZAMOJSKI

1. Introduction

Hyperspectral Imaging (HSI) collects and processes information from an as wide as possible spectrum of wavelengths, going far beyond the visible spectrum. If the human eye sees visible light in mostly three bands (red, green and blue), one can divide the whole spectrum of electromagnetic radiations' frequency into several hundreds to thousands of fine bands. Moreover, a single area is sampled many times for image correction to account for a variety of measurement errors. A single hyperspectral image can therefore be quite big in itself, 2-3 Gigabytes being not rare.

Transmission, storage and processing of such large sets of data is practically challenging. Spectrometers are mounted on aircrafts or satellites, neither of which has the appropriate computing power or storage to deal with HSI. Dimensionality reduction methods provide solutions to those difficulties. The airbourne device encodes the images via compressive sensing methods and send it to the ground where more performant computers are available for decoding and all the heavy lifting.

A recurrent paradigm in these compressive sensing methods is to represent images as matrices, compress them to lower-dimensional matrices, which are then factorized. The result is a low-rank matrix approximation to the original image matrix. In 2006, Martinsson, Rokhlin and Tygert published a new approach to probabilistic low-rank matrix approximations that we will refer to as randomized singular value decomposition (rSVD) [**?**]. A survey and extension on the subject can be found in the highly recommended seminal paper of Halko, Martinsson and Tropp [**?**]. Furthermore, Martinsson and Voronin provide an implementation of the algorithm with its description in [**?**], while Candès and Witten found a novel and intuitive analysis of the algorithm in [**?**]. Finally, rSVD was applied to HSI in [**?**] and compared to another probalistic method, compressive-projection principal component analysis (CPPCA).

Probabilistic methods to deal with hard problems have a long history, and it comes as no surprise that they provide powerful computational solutions in the world of Big Data. Statisticians, perhaps not so concerned with HSI, should nonetheless be interested in low-rank matrix approximation and rSVD as a performant solution to large principal component analysis (PCA).

Very popular amongst statisticians, the R language has many pros and cons. One of the cons we have encountered is that it is not designed for numerical linear algebra. Nonetheless, we would like to bring a practical solution in R for doing rSVD and programming numerical linear algebra in general. We note that an implementation using ordinary matrices has been provided in R through the package named rsvd. However, we would like to be able to handle matrices at a scale where one has to resort to dimension reduction for computational feasibility, where rSVD becomes very helpful.

Date: 30 March 2016.

The project thus consists in first installing a framework for efficient numerical linear algebra in R capable of working with large matrices allowing parallelisation as much as possible. We then implement rSVD in this framework and test its performance in the setting of image compressive sensing. Although HSI brings interesting challenges, it is mostly outside the scope of the current project. We rather restrict our focus to the usual color images in the visible spectrum. We nonetheless deal with a very large image, the sharpest take of the Andromede galaxy taken by the hubble telescope. The image is 4,5GB in size and is thus too large for R's memory restrictions. We compress the image using rSVD and compare its computational power and memory usage with other compression algorithms.

2. The Framework

- Parallelization Snow package...
- Memory R's bigmemory package ... Disappointed with bigalgebra.
- Numerical Linear Algebra The efficient C++ library *Armadillo* provides an easy to use API similar to matlab, integrates with other great and common libraries such as LAPACK, BLAS, ATLAS and Intel MKL and integrates seamlessly with R via Rcpp and RcppArmadillo. This is why we chose to use Armadillo for matrix computations.

As for out-of-core storage and manipulations, we have